TRAINING MANUAL IN OPEN SOURCE PLATFORM: BIAS CORRECTION OF SATELLITE PRECIPITATION

Prepared by:

Begum Rabeya Rushi, Regional Science Associate, NASA/SERVIR-HKH Earth System Science Center, University of Alabama, Huntsville.

Statement of Collaboration

Bias correction of Satellite Precipitation Product that are described in this training manual have been applied by SWAAT Project Research Group at the University of Arizona, Tucson. For this training purpose, all available resources have been recoded and translated into open source programming platform by the SERVIR Science Coordination Office (SCO). Faith Mitheu, Thematic Lead, Water Resources & Disasters at the Regional Centre for Mapping of Resources for Development (RCMRD), SERVIR-Eastern and Southern Africa, and Stella Masese, Technical and Logistics Assistant at the RCMRD, SERVIR-Eastern and Southern Africa supported the co-ordination effort to organize the bias correction of satellite products training in Nairobi, Kenya. Lee Ellenburg, Regional Science Coordination Lead, SERVIR Hindu Kush-Himalaya, Eric Anderson Regional Science Coordination Lead, SERVIR Mekong, Emily Adams, Regional Science Associate, SERVIR Eastern and Southern Africa, Africa Flores, Regional Science Coordination Lead, SERVIR Eastern and Southern Africa, and Kel Markert, Regional Science Associate, SERVIR Mekong provided time-to-time feedback to improve the training materials.

1. Introduction

Bias correction for satellite precipitation product (SPP) open source codes have been written in R/R Studio. We will adopted two different techniques for bias correction satellite precipitation methods:

a. Linear Scaling,
b. Quantile Mapping Technique

The methods described here were used in algorithm (written in MATLAB) developed by the SWAAT research group at the University of Arizona (Lead by Professor Juan Valdes) (Roy et al. 2016). For this training, algorithms for bias correction of SPP have been translated in R. R is an open source programming language and R Studio is the software environment used here and has separate versions for Linux/MAC/Windows users. R studio for desktop is universal and open source software for R programming environment.

Scripts for bias correction takes input from command line about input and output directory (Steps provided in Chapter 4). Detailed description about the scripts are provided in Chapter 5 for further analysis, if any user is interest.

2. R and R Studio Installation

Please follow the steps below for R and R Studio installation:

Windows

If you already have R and RStudio installed

- Open RStudio, and click on “Help” > “Check for updates”. If a new version is available, quit RStudio, and download the latest version for RStudio.
- To check the version of R you are using, start RStudio and the first thing that appears on the terminal indicates the version of R you are running. Go on the CRAN website and check whether a more recent version is available. If so, please download and install it. You may also want to consider removing your old version of R. You can check here for more information.

If you don’t have R and RStudio installed

- Download R from the CRAN website.
- Run the .exe file that was downloaded
- Go to the RStudio download page

- Under *Installers* select **RStudio x.yy.zzz - Windows XP/Vista/7/8** (where x, y, and z represent version numbers)
- Double click the file to install it
- Once it’s installed, open RStudio to make sure it

MacOS X

If you already have R and RStudio installed

- Open RStudio, and click on “Help” > “Check for updates”. If a new version is available, quit RStudio, and download the latest version for RStudio.
- To check the version of R you are using, start RStudio and the first thing that appears on the terminal indicates the version of R you are running. Go on the CRAN website and check whether a more recent version is available. If so, please download and install it. You may also want to consider removing your old version of R. You can check here for more information.

If you don’t have R and RStudio installed

- Download R from the CRAN website.
- Select the .pkg file for the version of OS X that you have and the file will download
- Double clik on the downloaded file to install R
- Go to the RStudio download page
- Under *Installers* select **RStudio x.yy.zzz - Mac OS X 10.6+ (64-bit)** (where x, y, and z represent version numbers)
- Double click the file to install RStudio
- Once it’s installed, open RStudio to make sure it works

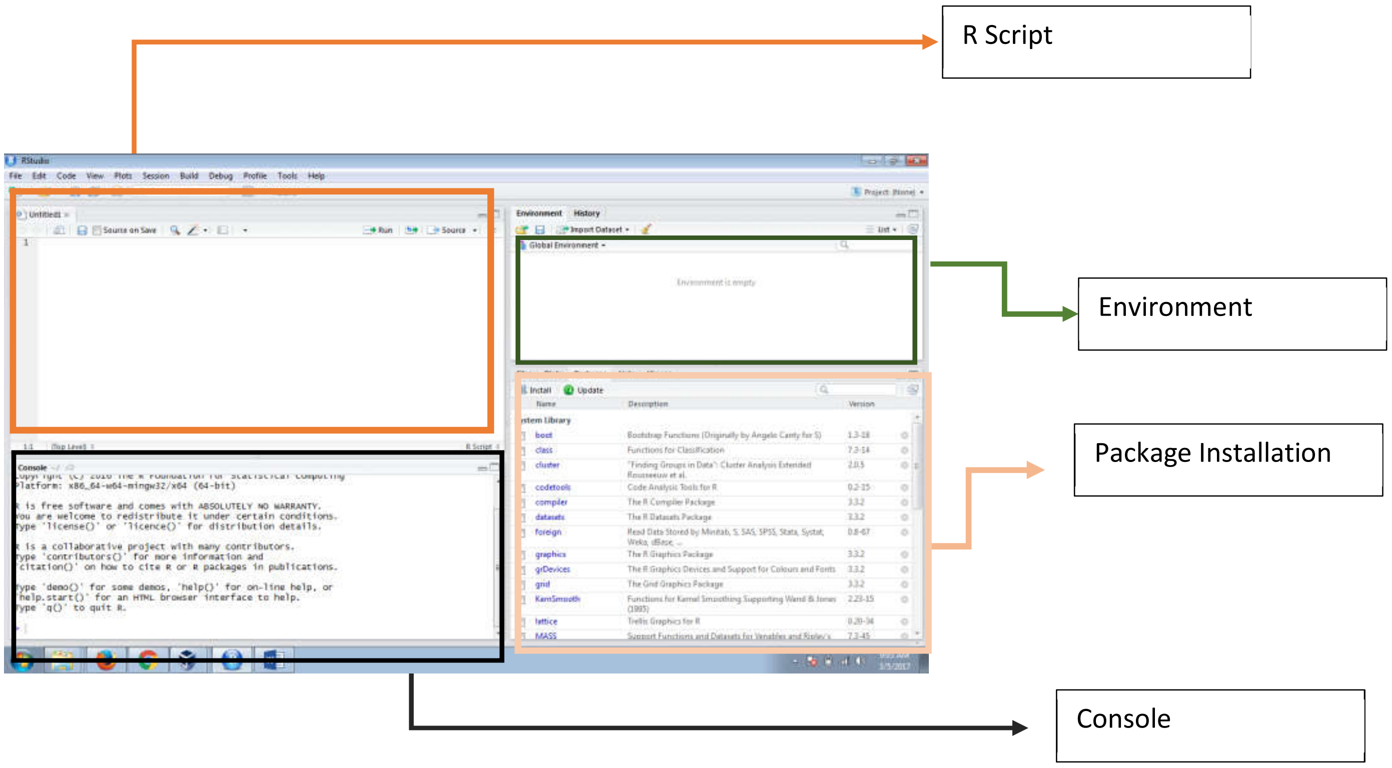
Linux

- Follow the instructions for your distribution from CRAN, they provide information to get the most recent version of R for your distribution. For most distributions, you could use your package manager (e.g., for Debian/Ubuntu run sudo apt-get install r-base, and for Fedora sudo yum install R), but the versions provided by this approach are usually out of date. In any case, make sure you have at least R 3.3.1
- Go to the RStudio download page

- Under *Installers* select the version that matches your distribution, and install it with your preferred method (e.g., with Debian/Ubuntu sudo dpkg -i rstudio-x.yy.zzz-amd64.deb at the terminal).
- Once it's installed, open RStudio to make sure it works

2. Package Installation and Data Preprocessing

2.1 Package Installation



```
Console ~/
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Name	Description	Version
System Library		
boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-18
class	Functions for Classification	7.3-14
cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.0.5
codetools	Code Analysis Tools for R	0.2-15
compiler	The R Compiler Package	3.3.2
datasets	The R Datasets Package	3.3.2
foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...	0.8-67
graphics	The R Graphics Package	3.3.2
grDevices	The R Graphics Devices and Support for Colours and Fonts	3.3.2
grid	The Grid Graphics Package	3.3.2
KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)	2.23-15
lattice	Trellis Graphics for R	0.20-34
MASS	Support Functions and Datasets for Venables and Ripley's	7.3-45

Figure 3.1: R Studio

R Studio has four segments/sections (Figure 3.1). After downloading R and R Studio, click "Install" under Package Installation section and search for necessary packages. To run the bias correction scripts, the following packages are needed:

1. Raster	4. abind	7. EDISON
2. sp	5. MASS	8. MCMCpack
3. rgdal	6. Pscl	9. outparse

Or, in console section you can write:

```
install.packages(c("package name"))
```

After installing all packages your bias correction model is ready to run.

2.2 Data Preprocessing:

The data used in this script should be located using the following formats:

- CHIRPS data should be organized as "~/chirps/Year/YYYY.mm.dd.tif"

  Example: C:/bias_correction/chirps/2015/2015.01.01.tif

- Rain gauge data should be organized as "~/observed/STATION NAME/Year/YYYY_daily.csv

  Example: C:/bias_correction/observed/BOGRA/2015/2015_daily.csv

  In the files, data should be organized as:

Date	Lat	Lon	PRCP
20150101	23.883	91.25	0
20150102	23.883	91.25	166.65
20150103	23.883	91.25	16.65

  Where PRCP is Precipitation in millimeters.

- Latitude and Longitude values of Rain Gauge Locations should be in a csv file and located as ~/NCDC_point_available.csv

  Example: C:/ NCDC_point_available.csv

  In the files, data should be organized as:

station	Lat	Lon
SAIDPUR	25.75	88.917
RANGPUR	25.733	89.233
DINAJPUR	25.65	88.683

- SPP data should be organized as "~/SPP/Year/YYYY.mm.dd.tif"

  Example: C:/bias_correction/persian/2015/2015.01.01.tif

3. Steps for Running the Scripts:

3.1 Linear Scaling Method

3.1.1 Step 1: Correcting CHIRPS using Point Location Data

Using Linear_Method_SPP_bias_correction_a.R, CHIRPS precipitation can be bias corrected using point based ground observations.

Using the command line, inputs are CHIRPS, latitude and lognitude information of all gauge locations, and rain gauge data. Also, the start and end year for running the model are needed to be defined as integers.

-i : Ingest directory of folder for CHIRPS precipitation data

-l: Ingest CSV file with Latitude and Longitude of all locations of gauges

-o: Ingest directory of observed data (rain gauge) folder

-Y: Start Year (integer)

-N: End Year (integer)

Outputs are comparison between CHIRPS and observed data and rain gauge location (format CSV files) and corrected CHIRPS (format geoTiff).

-p: Ingest directory of folder for storing comparison data between CHIRPS and rain gauge data

-c: Ingest directory of folder with corrected CHIRPS

Rscript --vanilla Linear_Method_SPP_bias_correction_a.R -i directory of input (CHIRPS) folder -l csv file with Latitude and Longitude of all locations of gauges -o directory of observed data (rain gauge) folder -Y Start Year -N End Year -p directory of folder for storing comparison data between CHIRPS and rain gauge data -c directory of folder with corrected CHIRPS

Example:

Rscript –vanilla Linear_Method_SPP_bias_correction_a.R –i C:/bias_correction/CHIRPS/
-l C:/bias_correction/NCDC_point_available.csv -o C:/bias_correction/observed/
-Y 2015 –N 2016 –p C:/bias_correction/comparison_CHIRPS_observed/ -c
C:/bias_correction/corrected_CHIRPS/

3.1.2 Step 2: Correcting SPP using Corrected CHIRPS

Using the Linear_Method_SPP_bias_correction_b.R script, SPP precipitation can be corrected using CHIRPS or Corrected CHIRPS.

Inputs are CHIRPS and raw SPP data. Also, the start and end year for running the model are needed to be defined as integers.

-o : Ingest directory of folder for CHIRPS/corrected precipitation data

-i: directory of input data (SPP)

-Y: Start Year (integer)

-N: End Year (integer)

Outputs are the corrected SPP data.

-c: Ingest directory of folder with corrected SPP data

Rscript --vanilla Linear_Method_SPP_bias_correction_b.R - o directory of observed data (CHIRPS/Corrected CHIRPS) folder -i directory of input data (SPP) -Y Start Year -N End Year -c directory of folder with corrected CHIRPS

Example:

Rscript –vanilla Linear_Method_SPP_bias_correction_b.R -o C:/bias_correction/CHIRPS/ -i C:/bias_correction/persian/ -Y 2015 –N 2016 -c C:/bias_correction/corrected_SPP/

4.2 Quantile Mapping Method

Script (Quantile_SPP_bias_correction.R) for quantile mapping method is only applicable for grided datasets. If rain gauge data are available in grid format, they can be used to correct the CHIRPS. Otherwise, using this code SPP precipitation can be corrected considering CHIRPS as the base product.

Here, inputs are CHIRPS and raw SPP data. Also, start and end year for running the model are needed to be defined as integer.

-o : Ingest directory of folder for CHIRPS/corrected precipitation data

-i: directory of input data (SPP)

-Y: Start Year (integer)

-N: End Year (integer)

On the other hand, outputs are corrected SPP data.

-c: Ingest directory of folder with corrected SPP data

Correct SPP using CHIRPS:

Rscript --vanilla Quantile_SPP_bias_correction_b.R - o directory of observed data (CHIRPS/Corrected CHIRPS) folder -i directory of input data (SPP) -Y Start Year -N End Year -c directory of folder with corrected CHIRPS

Example:

Rscript –vanilla Quantile_SPP_bias_correction_b.R -o C:/bias_correction/CHIRPS/
-i C:/bias_correction/persian/ -Y 2015 –N 2016 -c C:/bias_correction/corrected_SPP/

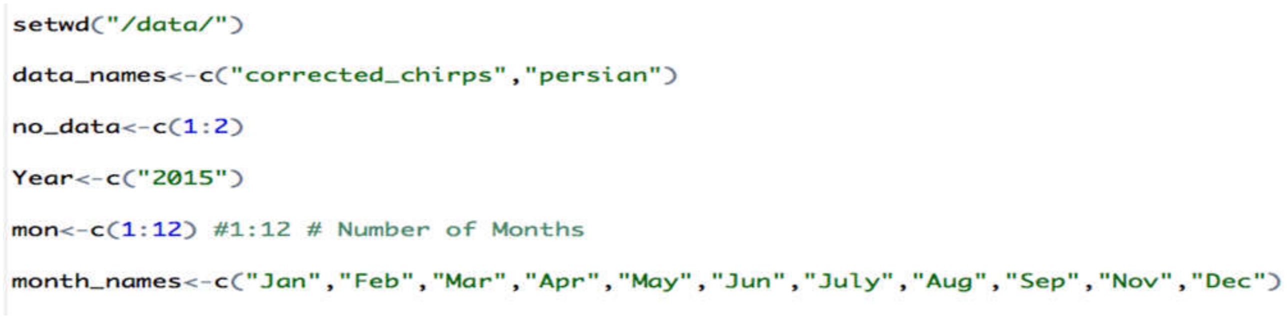
4. **Description of Bias Correction Methods:**

5.1 Linear Scaling Method:

The first bias correction method described will be the linear scaling method. Here we will take two gridded datasets, a 'truth" dataset, that should have minimial bias and a SPP product that we will bias correct. In this example, we will use CHIPRS 2.0 data as our 'truth' data to bias correct PERSIANN data.

Step 1: Create our global variables for the script:

First, we will set our initial working directory setwd("~/Linear_method"). Then we will create names for our two datasets to be used (***data_names<-c("corrected_chirps","persian"***). Next, we will define the year(s) we are working with (***Year***), the number of months (***mon***) and the month names (***month_names***). These variables will be called later in the script.



```
setwd("/data/")
data_names<-c("corrected_chirps","persian")
no_data<-c(1:2)
Year<-c("2015")
mon<-c(1:12) #1:12 # Number of Months
month_names<-c("Jan","Feb","Mar","Apr","May","Jun","July","Aug","Sep","Nov","Dec")
```

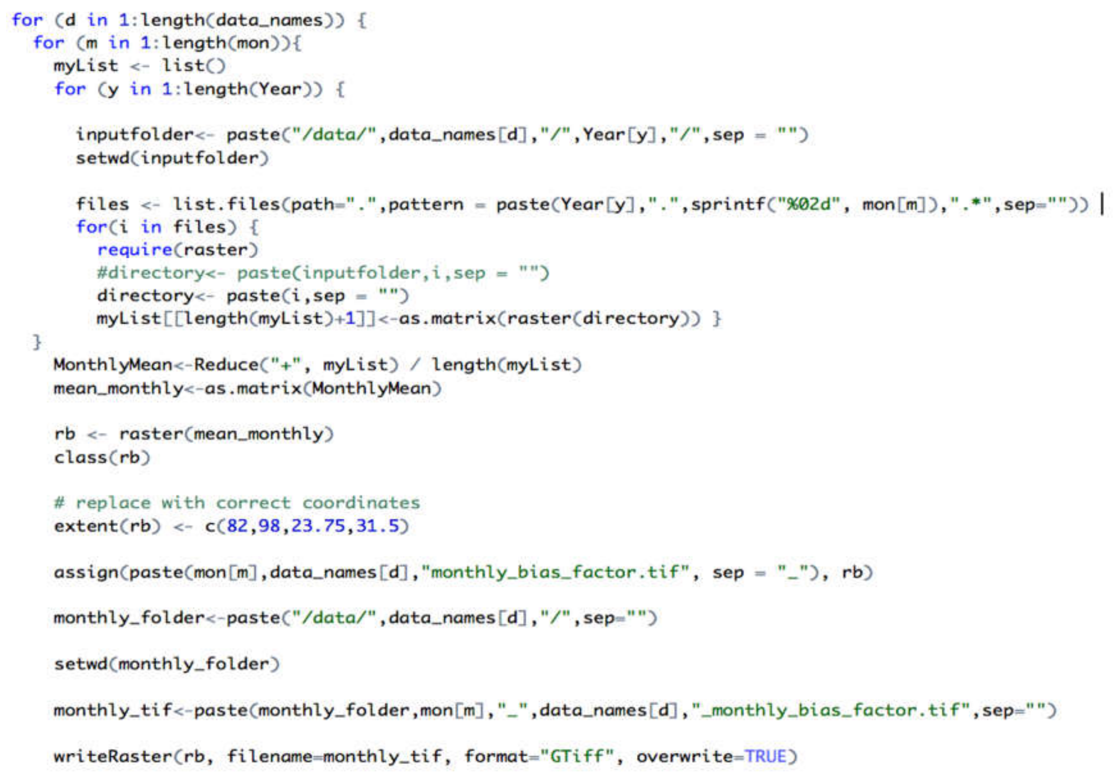
Step 2: Create monthly means

In this step we will loop through each GeoTiff (day) of each dataset (CHIRPS and PERSIANN) by month to create monthly means across all years per pixel and write them to an R matrix (similar to an array) for processing.

Sudo code:

```
for each dataset (CHIPRS and PERSIANN):
    for each month (1-12):
        for each year:
            input directory = data directory
            for each tiff in input directory (each day):
                create collection of all the rasters in the month as R matrix
    create montly means and save out as rasters in the dataset parent directory
```

As seen in the code snippet below, the montly means are created using the ***Reduce*** function and then stored as a 2-D matrix (***as.matrix***) where the x axis denotes longitude and the y-axis latitude. The ***extent*** of the monthly matrix is then set to ensure the location information is the same as the orginal clipped files. Next, a name is assigned (***assign***) for the matrix using the month (***mon***) and ***data_names*** variables. Lastly newly created monthly mean matrix (***monthly_tif***) is then saved out as a raster in the parent dataset directory to be accesed in the bais correction algorithm:



```
for (d in 1:length(data_names)) {
  for (m in 1:length(mon)){
    myList <- list()
    for (y in 1:length(Year)) {

      inputfolder<- paste("/data/",data_names[d],"/",Year[y],"/",sep = "")
      setwd(inputfolder)

      files <- list.files(path=".",pattern = paste(Year[y],".",sprintf("%02d", mon[m]),".*",sep=""))
      for(i in files) {
        require(raster)
        #directory<- paste(inputfolder,i,sep = "")
        directory<- paste(i,sep = "")
        myList[[length(myList)+1]]<-as.matrix(raster(directory)) }
  }
    MonthlyMean<-Reduce("+", myList) / length(myList)
    mean_monthly<-as.matrix(MonthlyMean)

    rb <- raster(mean_monthly)
    class(rb)

    # replace with correct coordinates
    extent(rb) <- c(82,98,23.75,31.5)

    assign(paste(mon[m],data_names[d],"monthly_bias_factor.tif", sep = "_"), rb)

    monthly_folder<-paste("/data/",data_names[d],"/",sep="")

    setwd(monthly_folder)

    monthly_tif<-paste(monthly_folder,mon[m],"_",data_names[d],"_monthly_bias_factor.tif",sep="")

    writeRaster(rb, filename=monthly_tif, format="GTiff", overwrite=TRUE)
```

Step 3: Perform bias correction of PERSIANN data using CHIRPS

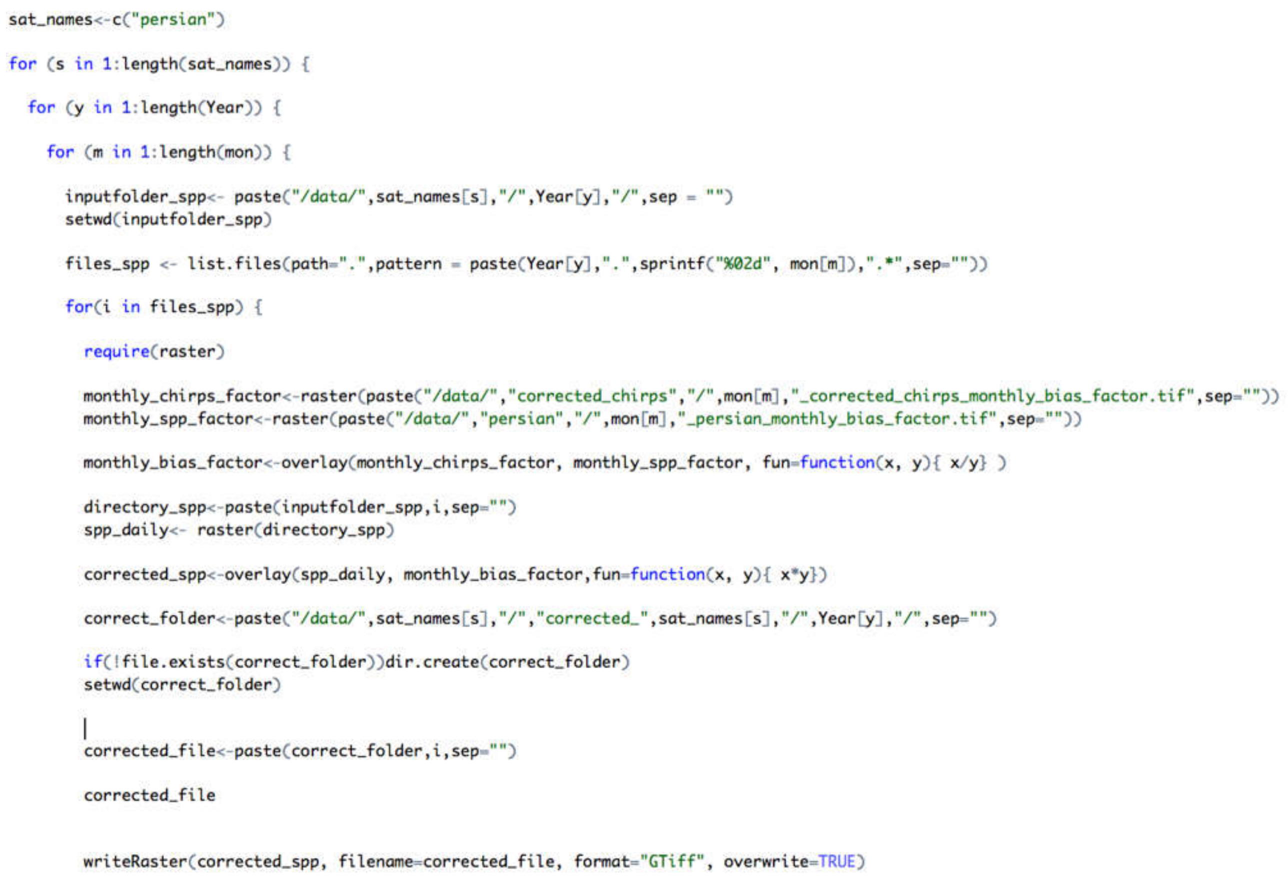
Finally, now that we have montly mean rasters we will calculate the bias of the PERSIANN dataset as compared to the CHIRPS and develop a linear correction factor that will be used to correct the PERSIANN data.

Sudo code:

```
for each dataset to be corrected:
    for each year:
        for each month:
            for each day in the month:
                define the monthly means
                calculate the monthly correction factor
                create corrected data
                export to corrected folder
```

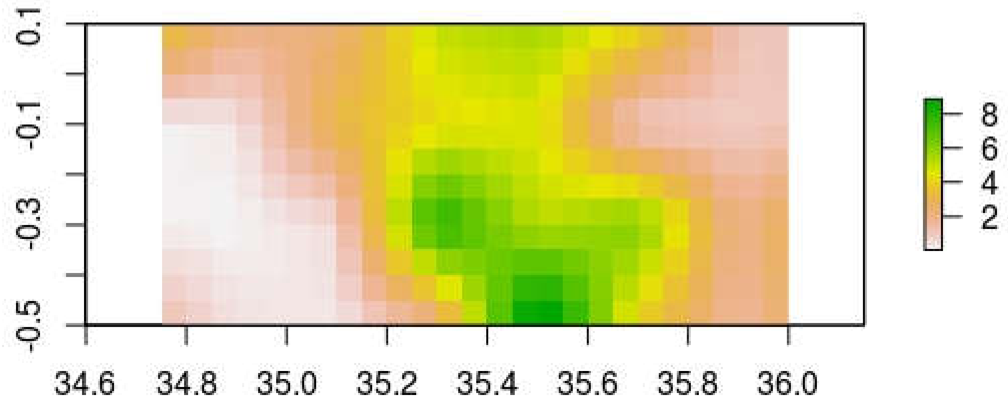
Code Description:

First, we loop through each dataset to be corrected (***sat_names***, here we are just using one, PERSIANN), year (***Year, y***) and month (***mon, m***). Next, we define an input folder for the PERSIANN data (inputfolder_spp), list all the daily tiffs in the directory (***list.files***) and begin to loop though each daily PERSIANN (***for (i in files_spp***). We then bring in each monthly mean dataset created in the step 2 above (***monthly_chirps_factor*** and ***monthly_spp_factor***).
Now using the ***overlay*** function in R, we calculate the monthly bias correction factor (***monthly_bias_factor***) by dividing the monthly mean for the CHIPRS data by the monthly mean for the PERSIANN data. This factor is then multiplied by each daily PERSIANA file in the month the current loop is in (Equation 1) to create the corrected PERSIANN data. Lastly, we define a correction folder (***correct_folder)*** and export the final corrected raster as a TIFF. A check is applied (***file.exist***) to ensure the output directory exists. In not one is created (***dir.create***)under the parent dataset drectory.

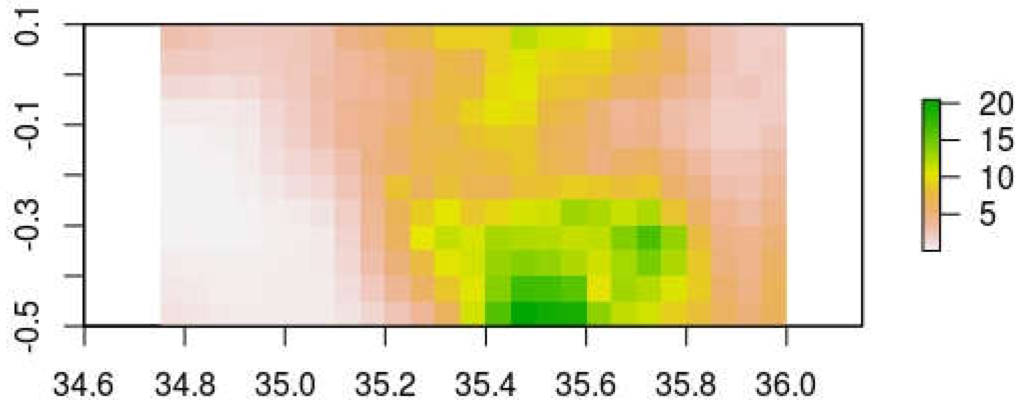


```
sat_names<-c("persian")

for (s in 1:length(sat_names)) {

  for (y in 1:length(Year)) {

    for (m in 1:length(mon)) {

      inputfolder_spp<- paste("/data/",sat_names[s],"/",Year[y],"/",sep = "")
      setwd(inputfolder_spp)

      files_spp <- list.files(path=".",pattern = paste(Year[y],".",sprintf("%02d", mon[m]),".*",sep=""))

      for(i in files_spp) {

        require(raster)

        monthly_chirps_factor<-raster(paste("/data/","corrected_chirps","/",mon[m],"_corrected_chirps_monthly_bias_factor.tif",sep=""))
        monthly_spp_factor<-raster(paste("/data/","persian","/",mon[m],"_persian_monthly_bias_factor.tif",sep=""))

        monthly_bias_factor<-overlay(monthly_chirps_factor, monthly_spp_factor, fun=function(x, y){ x/y} )

        directory_spp<-paste(inputfolder_spp,i,sep="")
        spp_daily<- raster(directory_spp)

        corrected_spp<-overlay(spp_daily, monthly_bias_factor,fun=function(x, y){ x*y})

        correct_folder<-paste("/data/",sat_names[s],"/","corrected_",sat_names[s],"/",Year[y],"/",sep="")

        if(!file.exists(correct_folder))dir.create(correct_folder)
        setwd(correct_folder)

        |
        corrected_file<-paste(correct_folder,i,sep="")

        corrected_file


        writeRaster(corrected_spp, filename=corrected_file, format="GTiff", overwrite=TRUE)
```

5.2 Visualization of Data:



5.1 Before Bias Correction



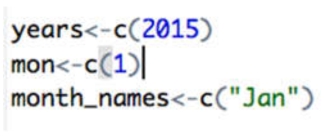
5.2 After Bias Correction

6. Quantile Mapping Method:

Similar to the linear method in section 1, here we will take two gridded datasets, a 'truth" dataset, that should have minimal bias and a SPP product that we will correct. In this example, we will use CHIPRS 2.0 data as our 'truth' data to bias correct PERSIANN data. However, instead of calculating monthly means, we are fitting a gamma distribution across all days, across all years, for a single month.

<u>Step 1: Create our global varaibles for the script:</u>

First will define the year(s) we are working with (***Year***), the number of months (***mon***) and the month names (***month_names***). These vaiable will be called later in the script.



```
years<-c(2015)
mon<-c(1)
month_names<-c("Jan")
```

Step 2: Begin month and year loops and collect data as 3-D monthly matrix

In this method, all data are designed to be loaded under two loops. The outer loop is month and inner loop is year.

Sudo code:

```
for each each month:
    for each year:
        generate list of daily files for each data
        for each file in CHIRPS daily list:
            create daily raster collection as 3-D matrix
        for each file in PERSIANN daily list:
            create daily raster collection as 3-D matrix
```

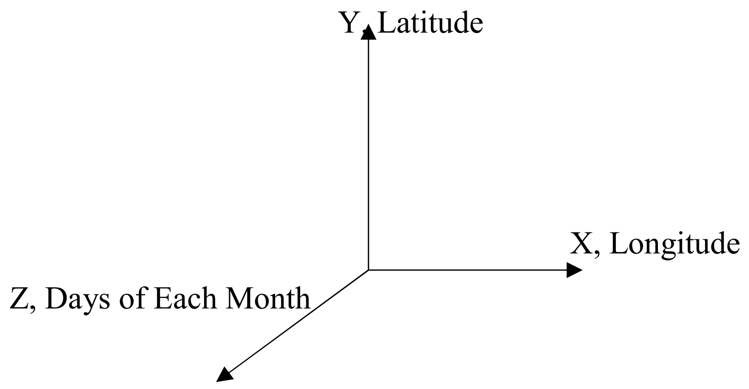


Figure 6.1: The 3-D Matrix for Each Month Considering All Years

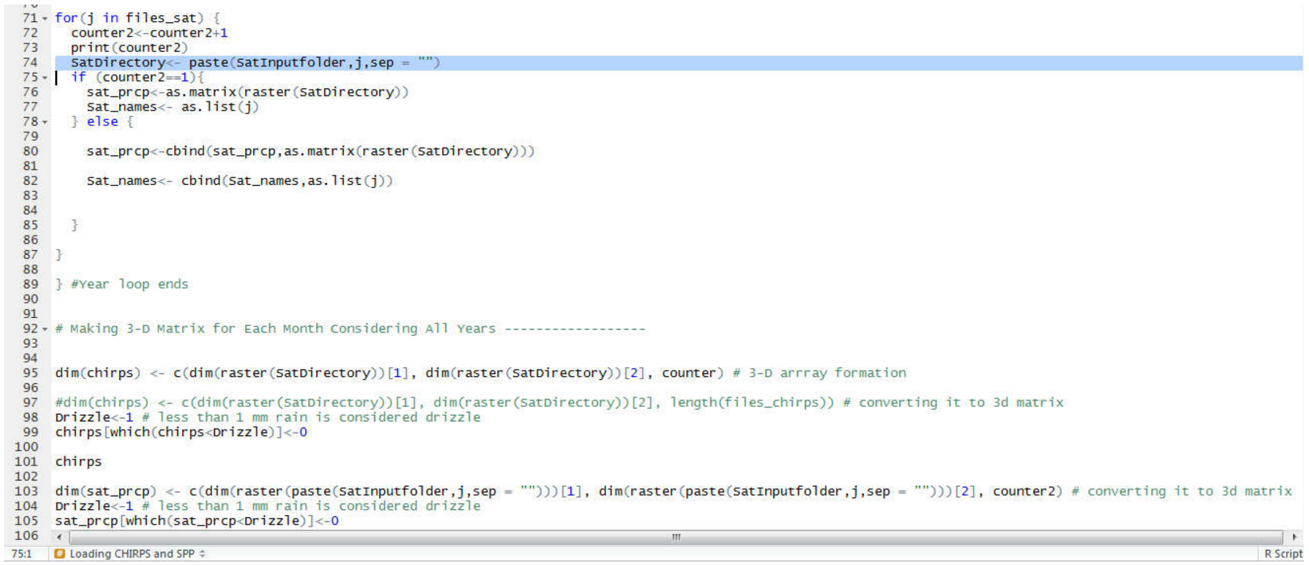
Code Description:

In this step a loop is first defined by up defined by the number of months set in the mon (***m, length(years***) variable above, then for each year (***y***, ***length(years)).***

Next, the data directories are defined (***ObsInputfolder*** and ***SatInputfolder***) for both CHIPRS and the satellite data to be corrected, respectively, then a list of all daily files in each directory is created (***files_chirps*** and ***files_sat***).

Each list is then looped though to create a time series of monthly data (3-D – x, y, t) as a R matrix for each dataset (***chirps*** and ***sat_prcp***). In this case, for each month we get one 3-D matrix. For example, if we are working with 2 years of data, for January along Z axis we will get 31 X 2 = 62 values (figure x).

Code Snippet



```r
for(j in files_sat) {
  counter2<-counter2+1
  print(counter2)
  SatDirectory<- paste(SatInputfolder,j,sep = "")
  if (counter2==1){
    sat_prcp<-as.matrix(raster(SatDirectory))
    Sat_names<- as.list(j)
  } else {

    sat_prcp<-cbind(sat_prcp,as.matrix(raster(SatDirectory)))

    Sat_names<- cbind(Sat_names,as.list(j))


  }

}

} #Year loop ends


# Making 3-D Matrix for Each Month Considering All Years ------------------


dim(chirps) <- c(dim(raster(SatDirectory))[1], dim(raster(SatDirectory))[2], counter) # 3-D arrray formation

#dim(chirps) <- c(dim(raster(SatDirectory))[1], dim(raster(SatDirectory))[2], length(files_chirps)) # converting it to 3d matrix
Drizzle<-1 # less than 1 mm rain is considered drizzle
chirps[which(chirps<Drizzle)]<-0

chirps

dim(sat_prcp) <- c(dim(raster(paste(SatInputfolder,j,sep = "")))[1], dim(raster(paste(SatInputfolder,j,sep = "")))[2], counter2) # converting it to 3d matrix
Drizzle<-1 # less than 1 mm rain is considered drizzle
sat_prcp[which(sat_prcp<Drizzle)]<-0
```

Step 3: Create 3-D matrices of gamma functions

Now that we have a time series for each month spanning all years considered, we will fit a gamma probability density function (PDF) to calculate the two parameters of the distribution (gamma (λ), and theta (θ)); stored in two separate matrices.

Sudo Code:

get dimension of month matirix for CHIRPS and PERSAIN

clean up values less than 1 (drizzle)

create cumulative density function (CDF) for each matrix

Step 4: Map CHIPRS Probability Distribution Function (PDF) parameters to PERSIAN

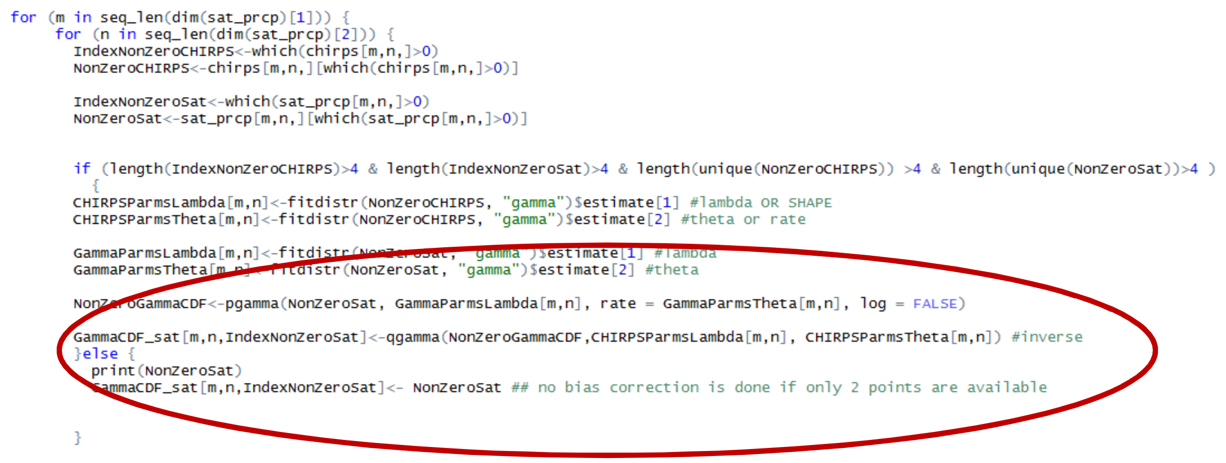
Sudo Code:

```
        for each month:
            for each day in the month:
                Extracting all non-zero values for CHIRPS
            Extracting all non-zero values for Satellite Data
    If there are more than 4 values in non-zero products:
            Fit those values and get parameters as λ, θ for CHIRPS
            Fit those values and get parameters as λ, θ for Satellite
          Determine probability value by using p gamma function λ, θ for Satellite data
        Determine Quantile value of Satellite by using q gamma function with lamda
    and theta of CHIRPS

    Else keep all values as it is
```

Code Description

Once we have the gamma and theta matrices for storing, we need to calculate one inverse function of Gamma-PDF to calculate probability (p gamma) for Satellite data and one forward function to get the bias corrected data using CHIRPS parameters.

Code Snippet



```
for (m in seq_len(dim(sat_prcp)[1])) {
    for (n in seq_len(dim(sat_prcp)[2])) {
      IndexNonZeroCHIRPS<-which(chirps[m,n,]>0)
      NonZeroCHIRPS<-chirps[m,n,][which(chirps[m,n,]>0)]

      IndexNonZeroSat<-which(sat_prcp[m,n,]>0)
      NonZeroSat<-sat_prcp[m,n,][which(sat_prcp[m,n,]>0)]


      if (length(IndexNonZeroCHIRPS)>4 & length(IndexNonZeroSat)>4 & length(unique(NonZeroCHIRPS)) >4 & length(unique(NonZeroSat))>4 )
        {
      CHIRPSParmsLambda[m,n]<-fitdistr(NonZeroCHIRPS, "gamma")$estimate[1] #lambda OR SHAPE
      CHIRPSParmsTheta[m,n]<-fitdistr(NonZeroCHIRPS, "gamma")$estimate[2] #theta or rate

      GammaParmsLambda[m,n]<-fitdistr(NonZeroSat, "gamma")$estimate[1] #lambda
      GammaParmsTheta[m,n]<-fitdistr(NonZeroSat, "gamma")$estimate[2] #theta

      NonZeroGammaCDF<-pgamma(NonZeroSat, GammaParmsLambda[m,n], rate = GammaParmsTheta[m,n], log = FALSE)

      GammaCDF_sat[m,n,IndexNonZeroSat]<-qgamma(NonZeroGammaCDF,CHIRPSParmsLambda[m,n], CHIRPSParmsTheta[m,n]) #inverse
      }else {
        print(NonZeroSat)
        GammaCDF_sat[m,n,IndexNonZeroSat]<- NonZeroSat ## no bias correction is done if only 2 points are available


      }
```

Step 4: Loop though each daily corrected daily PERSIANN file and export to GeoTIFF

Sudo code:

```
for each number of Geotiff per month:
        Save each (x,y) part of 3-D array as matrix
        Convert it to raster
        Define area extent
       Save as Geotiff file
```

Code Description

We saved numbers of Satellite Geotiff files available for each month as “counter2”. Now “corrected_spp_daily” is storing bias corrected precipitation for each data from the 3-D array after bias correction. Then the matrix is converted to a raster. After that, the area extent is defined and exported to Geotiff and saved.

Code snippet

```r
for (kk in 1:counter2){
print(kk)
 corrected_spp_daily <- as.matrix(GammaCDF_sat[,,kk])

 rb <- raster(corrected_spp_daily)
 class(rb)

 # replace with correct coordinates
 extent(rb) <- c(82,98,23.75,31.5)


 correct_folder<-paste("C:/chirps/quantile/Brahmaputra/corrected_persian/",sep="")

# if(!file.exists(correct_folder))dir.create(correct_folder)

 correct_spp_file<-paste(correct_folder,Sat_names[kk],sep="")
print(correct_spp_file)

 writeRaster(rb, filename=correct_spp_file, format="GTiff", overwrite=TRUE)

}
```

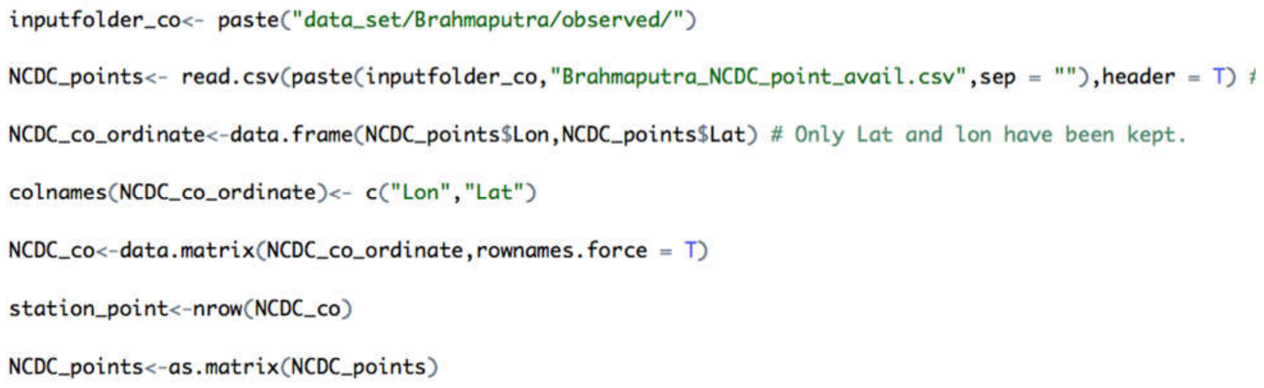
7. Bias Correction of gridded rainfall with station data

Here we will demonstrate how to use the linear method discussed above to bias correct a gridded rainfall product using available observed precipitation station data. In this example, we will use CHIPRS 2.0 and climate stations available from the US National Centers for Environmental Information (https://www7.ncdc.noaa.gov/CDO/cdoselect.cmd?datasetabbv=GSOD&countryabbv=&georegionabbv) to demonstrate.

Step 1: Import and clip station data

The first step is to get the area extent of the river basin to identify the relevant NCEI gauge locations. A list of the latest location of gauge stations can be found at ftp://ftp.ncdc.noaa.gov/pub/data/noaa/isd-history.txt

This step can be completed in ArcGIS using the “Clip” function. Here, it is important to mention that we will only consider those locations which have data throughout the study period.
For example: if our study period is from 1981 to 2010 (30 years), it will consider only those stations which have data for the entire 30 periods. In this training we will use one year, 2015, for demonstration purposes.
Code Snipit header?



```
inputfolder_co<- paste("data_set/Brahmaputra/observed/")

NCDC_points<- read.csv(paste(inputfolder_co,"Brahmaputra_NCDC_point_avail.csv",sep = ""),header = T) #

NCDC_co_ordinate<-data.frame(NCDC_points$Lon,NCDC_points$Lat) # Only Lat and lon have been kept.

colnames(NCDC_co_ordinate)<- c("Lon","Lat")

NCDC_co<-data.matrix(NCDC_co_ordinate,rownames.force = T)

station_point<-nrow(NCDC_co)

NCDC_points<-as.matrix(NCDC_points)
```

Once we have the locations of the gauging stations and their corresponding data set for the entire study period, we are ready to run the bias correction code written in R.

C:\Bias_correction\Nyando (basin name)\observed\(station name)\2015(year)\2015_daily.csv.

In the .csv file, data should be arranged:

Date	Lat	Lon	PRCP
20150101	23.883	91.25	0
20150102	23.883	91.25	16.665

Discussion on linear bias correction code:

<u>Step 1: Loading NCDC Data Points for the River Basin</u>

It is needed to load the file for the list of gauging stations that we made for the basin (using R).
How to load the file (add code here):

File should be arranged like this:

station	Lat	Lon
KERICHO	25.75	88.917

In the code developed by SCO, only latitude and longitude information are needed. Show code to do this here : inputfolder_co<- paste("data_set/Nyando/observed/")

Next, we read in the location information into a R data frame and a R matrix

Dataframe code?

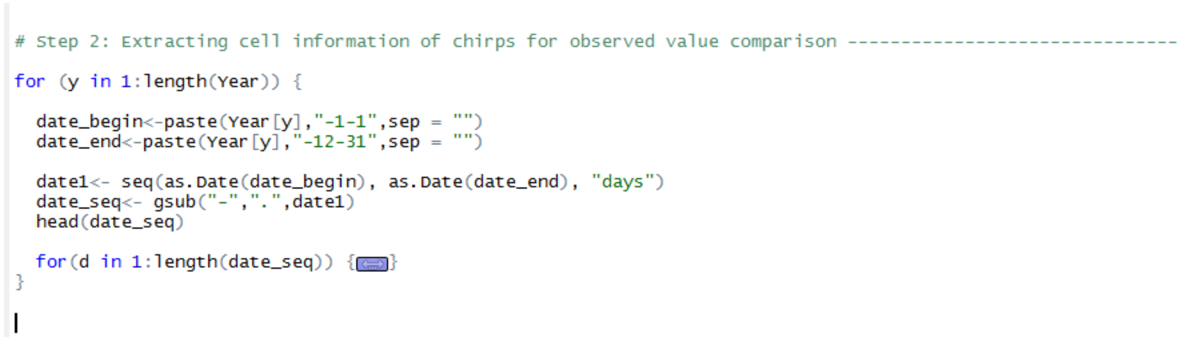
NCDC_points<-as.matrix(NCDC_points)

In order to loop though each available station, we need the number of station which is obtained using the following code:

station_point<-nrow(NCDC_co)

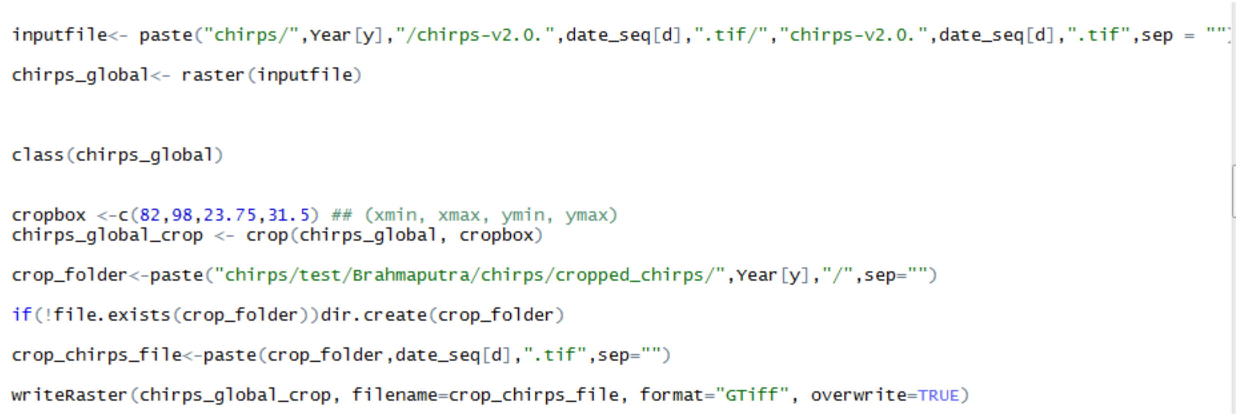
Step 2: Extracting cell information of CHIRPS for observed value comparison

Once we know the locations of the observation data, we can extract the corresponding CHIRPS cell values using latitude and longitude of those observation points. There are two major loops in this step (Figure 1.1). One loop organizes the year and another loop organizes date. If we consider two years (2015 -2016), at first we get all the daily values for 2015 and then will extract CHIRPS values for 2016.



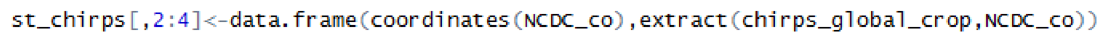
```
# Step 2: Extracting cell information of chirps for observed value comparison -------------------------------
for (y in 1:length(Year)) {

  date_begin<-paste(Year[y],"-1-1",sep = "")
  date_end<-paste(Year[y],"-12-31",sep = "")

  date1<- seq(as.Date(date_begin), as.Date(date_end), "days")
  date_seq<- gsub("-",".",date1)
  head(date_seq)

  for(d in 1:length(date_seq)) {}
}
```

In this step, we also crop the CHIRPS data according to area extent of the river basin and save it for future use.



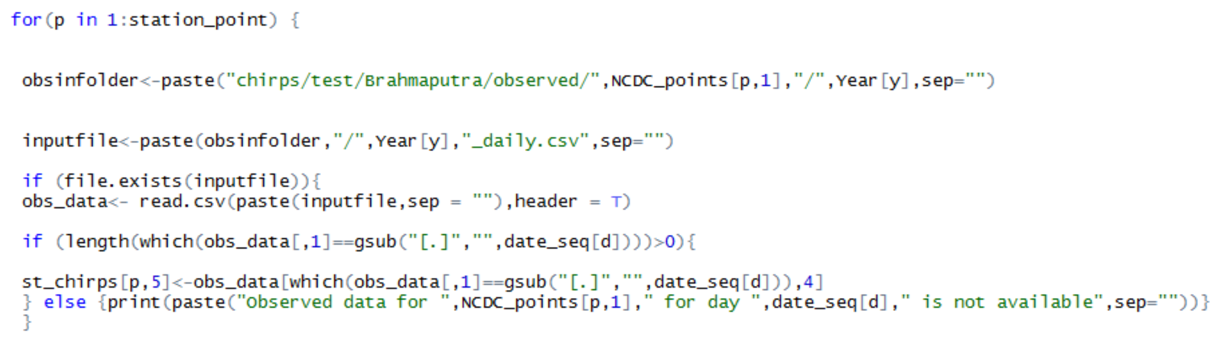
```
inputfile<- paste("chirps/",Year[y],"/chirps-v2.0.",date_seq[d],".tif/","chirps-v2.0.",date_seq[d],".tif",sep = ""
chirps_global<- raster(inputfile)


class(chirps_global)


cropbox <-c(82,98,23.75,31.5) ## (xmin, xmax, ymin, ymax)
chirps_global_crop <- crop(chirps_global, cropbox)

crop_folder<-paste("chirps/test/Brahmaputra/chirps/cropped_chirps/",Year[y],"/",sep="")

if(!file.exists(crop_folder))dir.create(crop_folder)

crop_chirps_file<-paste(crop_folder,date_seq[d],".tif",sep="")

writeRaster(chirps_global_crop, filename=crop_chirps_file, format="GTiff", overwrite=TRUE)
```

Once we have CHIRPS values for the river basin, we extract the corresponding CHIRPS cell values.



```
st_chirps[,2:4]<-data.frame(coordinates(NCDC_co),extract(chirps_global_crop,NCDC_co))
```

Sometimes, observation data are available for the entire study period; however, some daily values are missing. That is why we will consider the date for which both CHIRPS and observation data are available.



```
for(p in 1:station_point) {

 obsinfolder<-paste("chirps/test/Brahmaputra/observed/",NCDC_points[p,1],"/",Year[y],sep="")

 inputfile<-paste(obsinfolder,"/",Year[y],"_daily.csv",sep="")

 if (file.exists(inputfile)){
 obs_data<- read.csv(paste(inputfile,sep = ""),header = T)

 if (length(which(obs_data[,1]==gsub("[.]","",date_seq[d])))>0){

 st_chirps[p,5]<-obs_data[which(obs_data[,1]==gsub("[.]","",date_seq[d])),4]
 } else {print(paste("Observed data for ",NCDC_points[p,1]," for day ",date_seq[d]," is not available",sep=""))}
 }
```

Step 3: Monthly Component for Each Data Set (CHIRPS and Observation Data)

We know,

$$P_{\text{corrected, m,d}} = P_{\text{raw,m,d}} \times \frac{\mu\,(\text{Pobs,monthly})}{\mu\,(\text{Praw,monthly})}$$

where Pcorrected,m,d is corrected precipitation at given month m on dth day.

Praw,m,d is satellite precipitation at given month m on dth day. μ ($P_{\text{obs,monthly}}$) is the mean value of CHIRPS precipitation at given month m and μ ($P_{raw,monthly}$) is the mean value of satellite precipitation at given month m.

In Step 3, we will calculate mean monthly values for CHIRPS and observation data per pixel. It is to be noted that for each dataset will have a value for each month as a mean monthly component.

```
# Step 3: Monthly Mean Component ----------------------------------------------


mon<-c(1:12) #1:12 # Number of Months

month_names<-c("Jan","Feb","Mar","Apr","May","Jun","July","Aug","Sep","Nov","Dec")

for (m in 1:length(mon)) {
  myList <- list()

  for (y in 1:length(Year)) {

    inputfolder<- paste("C:/chirps/test/Brahmaputra/chirps_obs_daily_comparison/",Year[y],sep = "")
    setwd(inputfolder)

    files <- list.files(path=".",pattern = paste(Year[y],".",sprintf("%02d", mon[m]),".*",sep="")) #path=inputfolder,

    for(i in files) { myList[[length(myList)+1]]<-as.matrix(read.csv(i)) }


  }
```

Step 4: Correction of CHIRPS with Observed Precipitation

After calculating the monthly mean for each for each data set, we will calculate the monthly bias factor by dividing the monthly mean for observed data by monthly mean for CHIRPS data. Then, we will multiple each daily CHIRPS value with corresponding monthly bias factor. Now the bias corrected CHIRPS values would be saved in a GeoTiff format.

```
# Step 4: Correction of CHIRPS with observed Precipitation ----------------

for (y in 1:length(Year)) {

  for (m in 1:length(mon)) {

    inputfolder_chirps<- paste("C:/chirps/test/Brahmaputra/chirps/cropped_chirps/",Year[y],"/",sep = "")
    setwd(inputfolder_chirps)


    files_chirps <- list.files(path=".",pattern = paste(Year[y],".",sprintf("%02d", mon[m]),".*",sep="")) #path=inpu


     for(i in files_chirps) {

       require(raster)

       directory<- paste(inputfolder_chirps,i,sep = "")

       chirps_daily<- as.matrix(raster(directory))


      monthly_factor<- as.numeric(lapply(paste(mon[m],"monthly_obs_bias_factor", sep = "_"),get))

      corrected_chirps<-data.matrix(chirps_daily*monthly_factor,rownames.force = T)
```

Reference:

Roy, T., Serrat-Capdevila, A., Gupta, H. and Valdes, J., 2016. A platform for probabilistic multimodel and multiproduct streamflow forecasting. *Water Resources Research*.